

06/16/00
JCS812 U.S. PRO

06-19-00

A

Please type a plus sign (+) inside this box → ☐

PTO/SB/05 (4/98)
Approved for use through 09/30/2000. OMB 0851-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL (Only for new nonprovisional applications under 37 C.F.R. § 1.53(b))	Attorney Docket No. MS150658.1
	First Inventor or Application Identifier Alexander E. Mallet, et al.
	Title SYSTEM AND METHOD FOR PARALLEL...
	Express Mail Label No. EL550123757US

APPLICATION ELEMENTS See MPEP chapter 600 concerning utility patent application contents.	ADDRESS TO: Assistant Commissioner for Patents Box Patent Application Washington, DC 20231	
1. <input checked="" type="checkbox"/> * Fee Transmittal Form (e.g., PTO/SB/17) (Submit an original and a duplicate for fee processing)	5. <input type="checkbox"/> Microfiche Computer Program (Appendix)	
2. <input checked="" type="checkbox"/> Specification [Total Pages 22] (preferred arrangement set forth below) <ul style="list-style-type: none">- Descriptive title of the invention- Cross References to Related Applications- Statement Regarding Fed sponsored R & D- Reference to Microfiche Appendix- Background of the invention- Brief Summary of the invention- Brief Description of the Drawings (if filed)- Detailed Description- Claim(s)- Abstract of the Disclosure	6. Nucleotide and/or Amino Acid Sequence Submission (if applicable, all necessary) <ul style="list-style-type: none">a. <input type="checkbox"/> Computer Readable Copyb. <input type="checkbox"/> Paper Copy (identical to computer copy)c. <input type="checkbox"/> Statement verifying identity of above copies	
3. <input checked="" type="checkbox"/> Drawing(s) (35 U.S.C. 113) [Total Sheets 6]	ACCOMPANYING APPLICATION PARTS 7. <input checked="" type="checkbox"/> Assignment Papers (cover sheet & document(s)) 8. <input type="checkbox"/> 37 C.F.R. § 3.73(b) Statement <input type="checkbox"/> Power of Attorney (when there is an assignee) 9. <input type="checkbox"/> English Translation Document (if applicable) 10. <input type="checkbox"/> Information Disclosure Statement (IDS)/PTO-1449 <input type="checkbox"/> Copies of IDS Citations 11. <input type="checkbox"/> Preliminary Amendment 12. <input checked="" type="checkbox"/> Return Receipt Postcard (MPEP 503) (Should be specifically itemized) * Small Entity <input type="checkbox"/> Statement filed in prior application, Status still proper and desired (PTO/SB/09-12) 13. <input type="checkbox"/> Statement(s) 14. <input type="checkbox"/> Certified Copy of Priority Document(s) (if foreign priority is claimed) 15. <input checked="" type="checkbox"/> Other: Express Mail Certificate 37 CFR 1.10	
4. Oath or Declaration [Total Pages 3] <ul style="list-style-type: none">a. <input checked="" type="checkbox"/> Newly executed (original or copy)b. <input type="checkbox"/> Copy from a prior application (37 C.F.R. § 1.63(d)) (for continuation/divisional with Box 16 completed)<ul style="list-style-type: none">i. <input type="checkbox"/> DELETION OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 C.F.R. §§ 1.63(d)(2) and 1.33(b).		
NOTE FOR ITEMS 1 & 13: IN ORDER TO BE ENTITLED TO PAY SMALL ENTITY FEES, A SMALL ENTITY STATEMENT IS REQUIRED (37 C.F.R. § 1.27), EXCEPT IF ONE FILED IN A PRIOR APPLICATION IS RELIED UPON (37 C.F.R. § 1.28).		

16. If a CONTINUING APPLICATION, check appropriate box, and supply the requisite information below and in a preliminary amendment:
☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No: _____
Prior application information: Examiner _____ Group / Art Unit: _____
For CONTINUATION or DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 4b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

17. CORRESPONDENCE ADDRESS					
<input type="checkbox"/> Customer Number or Bar Code Label (Insert Customer No. or Attach bar code label here) or <input checked="" type="checkbox"/> Correspondence address below					
Name	Amin, Eschweiler & Turocy, LLP				
	Himanshu S. Amin				
Address	24th Floor, National City Center				
	1900 East 9th Street				
City	Cleveland	State	Ohio	Zip Code	44114
Country		Telephone	216-696-8730	Fax	216-696-8731

Name (Print/Type)	Himanshu S. Amin	Registration No. (Attorney/Agent)	40,894
Signature		Date	6/16/00

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231.

Atty. Docket No. MS150658.1

**SYSTEM AND METHOD FOR PARALLEL
ASYNCHRONOUS EXECUTION OF COMMANDS**

by

Alexander E. Mallet, Justin Grant, Michael W. Thomas

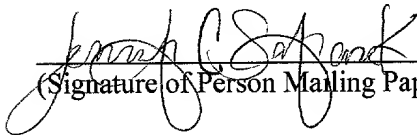
CERTIFICATION UNDER 37 CFR 1.10

I hereby certify that the attached patent application (along with any other paper referred to as being attached or enclosed) is being deposited with the United States Postal Service on this date **June 16, 2000**, in an envelope as "Express Mail Post Office to Addressee"

Mailing Label Number **EL550123757US** addressed to the: Box Patent Application, Assistant Commissioner for Patents, Washington, D.C. 20231.

Jennifer C. Safranek

(Typed or Printed Name of Person Mailing Paper)


(Signature of Person Mailing Paper)

**Title: SYSTEM AND METHOD FOR PARALLEL ASYNCHRONOUS
EXECUTION OF COMMANDS**

5

Technical Field

The present invention relates generally to computer systems, and more particularly to a system and method for executing asynchronous cluster commands in a distributed network environment.

10

Background of the Invention

As computing systems have become more dependent on network communications, system performance requirements have increasingly become more demanding. Many of these requirements are related to distributed system architectures wherein a first system, such as a client, may need to invoke remote command execution on one or more other systems such as server systems. In many instances, it is desirable to execute a plurality of commands on multiple servers concurrently – without having to wait for a command to complete on one server before moving to the next server. Waiting becomes even more important when commands require substantial time to complete. Unfortunately, conventional system architectures do not provide an acceptable interface to enable remote commands to execute substantially in parallel without undue delay. Thus, remote command performance needs to be improved in order to meet and/or exceed the challenges presented by modern distributed systems.

Another challenge associated with remote command execution is related to command execution during and/or after a disruption in communications between the client and the server. The disruptions may be caused by network problems (*e.g.*, intermittent connection) or may be hardware/software related whereby a server must be rebooted because of a system crash. In some instances, system commands may cause a disruption in network communications. For example, when an IP address is added to a network card, all existing TCP connections for that card may be dropped. Network disruptions, as described above, will likely present a problem for conventional

architectures. If the command were executed as part of a conventional remote procedure call, then the procedure will likely fail in the event of a disruption since these processes rely on TCP connections remaining open for the duration of the call.

Some attempts have been implemented to try and address the above problems.

- 5 One approach involves splitting the remote procedure calls into two method invocations – a “Begin” and “Finish” invocation. In a Begin invocation, input parameters are sent to the method without waiting for the execution of the actual call. When the caller desires to retrieve the results of the invocation, the Finish invocation is called wherein output parameters of the method may be filled in with the result of the execution. Although
- 10 concurrent execution of commands may be provided by the Begin and Finish invocation, other problems are still unresolved. For example, two separate versions of the interface for the Begin and Finish invocation must be provided in order to execute the method – a synchronous and asynchronous version. Secondly, the client must explicitly request for the results of the remote method invocation and risk blocking if the execution is not
- 15 completed.

- Another attempt to address the above problems has involved the utilization of callback pointers wherein the client is required to pass a callback pointer in the method invocation. Upon receipt of the call, a server may spin a new thread, then pass the callback pointer to the thread and return. The new thread may then call the client back on
- 20 the supplied callback pointer when processing is complete. However, this interface may fail if the server calls back on an interface pointer that is no longer valid due to the client shutting down and/or crashing. Additionally, significant overhead is involved with callback pointers since these pointers must be marshaled across various processes, machines and threads.

- 25 Unfortunately, neither of the above implementations solve the problem caused by network connections being reset as part of the remote command invocation since both implementations substantially rely on persistent and/or continuous network connections remaining available during remote command execution.

Summary of the Invention

The present invention relates to a system and method for executing parallel asynchronous commands in a multiple services environment. In accordance with the present invention, a plurality of commands may be concurrently executed without waiting
5 for the status and result of the command to be received. This enables a process to complete even if a network connection is reset and/or temporarily disabled during remote command invocation. In contrast to conventional systems which may rely on persistent (e.g., continuous) connections being available during command execution, the present invention may remotely invoke execution on a first system, and receive completed results
10 on a second system at a later time – without the need for persistent handshaking to complete execution. Thus, command execution may start and then complete after a disruption in network communications.

More specifically, the present invention provides a distributed object architecture that is managed *via* a thread pool of concurrently executable objects. Remote completion
15 of a command may then be signaled *via* an event fired from a remote machine. A remote invocation of a command may be started by calling the distributed object, and notification/results of the completion of the invocation may then be received as a result of the event fired from the remote machine. When a first system (e.g., client) makes a command call to the remote machine, the call returns before execution actually begins.
20 Thus, a plurality a remote commands may be executed in rapid succession. The first system may then release an interface pointer after making the initial command calls since notification of the calls is provided from the remote machine by the event. This mitigates the problem of a server calling back on an interface pointer that may no longer be valid due to a client being shut down and/or crashed. Moreover, overhead associated
25 with marshalling interface pointers across different processes, machines, and threads is mitigated.

According to another aspect of the present invention, system upgradability and flexibility may be improved by enabling a system to receive new functionality without disturbing and/or requiring existing system components to change. This may be achieved
30 by implementing an interface in accordance with the present invention, passing input data

to the interface in the form of a string and marshalling output data into an output argument to be received by the interface.

In accordance with another aspect of the present invention, a system is provided for parallel asynchronous command execution. The system includes a first computer
5 system for directing a call to invoke a remote procedure in a second computer system, wherein the first computer and second computer communicate *via* a non-persistent connection. The second computer system upon completion of the remote procedure generates an event trigger and transmits the event trigger and remote procedure results to the first computer.

According to yet another aspect of the present invention, a system is provided for parallel asynchronous command execution. The system includes a first computer system for directing at least one call to invoke a remote procedure in at least one other computer system, the at least one other computer system upon completion of the remote procedure establishing a non-persistent connection to the first computer system, the at least one
15 other computer system generating an event trigger and transmitting the event trigger and remote procedure results to the first computer.

According to another aspect of the present invention, a methodology is provided for parallel asynchronous command execution. The methodology includes the steps of: directing at least one call from a first computer system to invoke a remote procedure in at
20 least one other computer system; establishing a non-persistent connection between the first computer system and the at least one other computer system upon completion of the remote procedure; and generating an event trigger and transmitting the event trigger and remote procedure results to the first computer.

In accordance with another aspect of the present invention, a system is provided
25 for parallel asynchronous command execution. The system includes means for directing at least one call from a first computer system to invoke a remote procedure in at least one other computer system and means for establishing a non-persistent connection between the first computer system and the at least one other computer system upon completion of the remote procedure. The system also includes means for generating an event trigger
30 and transmitting the event trigger and remote procedure results to the first computer

system.

Another aspect of the present invention includes a system for parallel asynchronous command execution. The system includes a server for responding to at least one remote call by invoking a remote procedure, the server establishes a non-
5 persistent connection to communicate results of the remote procedure, and the server generates an event trigger and transmits the event trigger and remote procedure results upon completion of the remote procedure.

According to another aspect of the present invention, a system for parallel asynchronous command execution is provided. The system includes a first computer for
10 directing a call to invoke a remote procedure in a second computer, the first computer transmits a non-persistent signal to the second computer, wherein the second computer upon completion of the remote procedure generates an event trigger and transmits the event trigger and remote procedure results to the first computer *via* the signal.

In accordance with yet another aspect of the present invention, a computer-
15 readable medium is provided having computer-readable instructions for performing the acts of:
responding to at least one remote call by invoking a remote procedure; establishing a non-persistent connection to communicate results of the remote procedure; and generating an event trigger and transmitting the event trigger and remote procedure results upon
20 completion of the remote procedure.

To the accomplishment of the foregoing and related ends, the invention then, comprises the features hereinafter fully described. The following description and the annexed drawings set forth in detail certain illustrative aspects of the invention. These aspects are indicative, however, of but a few of the various ways in which the principles
25 of the invention may be employed and the present invention is intended to include all such aspects and their equivalents. Other advantages and novel features of the invention will become apparent from the following detailed description of the invention when considered in conjunction with the drawings.

30

Brief Description of the Drawings

Fig. 1 is a schematic block diagram illustrating a system for parallel remote command execution in accordance with one aspect of the present invention;

Fig. 2 is a schematic block diagram illustrating a system employing events and a distributed object architecture in accordance with one aspect of the present invention;

Fig. 3 is a flow chart diagram illustrating a methodology for client/server remote command execution in accordance with one aspect of the present invention;

Fig. 4 is a flow chart diagram illustrating a methodology for server execution in accordance with one aspect of the present invention;

Fig. 5 is a flow chart diagram illustrating a methodology for client execution in accordance with one aspect of the present invention; and

Fig. 6 is a schematic block diagram illustrating a system in accordance with one aspect of the present invention.

Detailed Description of the Invention

The present invention is now described with reference to the drawings, wherein like reference numerals are used to refer to like elements throughout.

In accordance with the present invention, a system and method is provided for executing concurrent asynchronous cluster commands – even if network connections are disrupted after being reset as part of remote command invocation. Commands may execute without waiting for previously initiated commands and do not rely on persistent network connections being available during command invocation. Thus, by enabling commands to be fired in rapid succession and by further enabling command completion after network disruptions have occurred, substantial performance improvements are achieved over conventional systems.

Moreover, the present invention provides an extensibility mechanism for adding desired functionality to a system and/or system cluster without disrupting existing system functionality. Extensibility is especially important in distributed systems wherein it may be infeasible to upgrade all nodes at once. Consequently, new system functionality may need to be introduced in a phased manner.

Referring initially to Fig. 1, a system 10a illustrates a particular aspect of the present invention relating to asynchronous execution of commands in a network environment. A first computer system 20 may direct a plurality of remote calls 24 to a plurality of remote systems 26 and 28. For example, remote system 1 (26) and remote system N 28 (N being any integer) may receive calls *via* a network 30. The calls 24 may be directed at particular points in time such as a first call 24a directed at time t1 and a second call 24b directed at time t2.

According to the present invention, the calls 24a and 24b may invoke a remote command execution on the systems 26 and 28 and receive a response 32a and 32b (e.g., data, flags, status) after the occurrence of a network disruption 34. The network disruption 34 may occur, for example, at times t1' and t2' and may result from network connections being reset in response to the remote calls 24. Thus, the present invention does not rely on persistent network connections remaining available in order to complete remote command execution.

As will be described in more detail below, the calls 24 may be initiated in a rapid manner. This may be achieved, for example, by enabling a calling function (not shown) to return within the computer system 20 without waiting for remote execution to begin. In accordance with the present invention, an Event 36a and 36b may be configured by the computer system 20 to receive notification of when remote execution is complete - thereby enabling the computer 20 to perform other tasks. Furthermore, the Events 36a and 36b enable the computer system 20 to execute a plurality of commands in parallel without waiting for results from any particular command.

Upon completion of the remote commands, an Event Trigger 32a' and/or 32b' may be initiated as a portion of the responses 32a and 32b from the remote systems 26 and 28. The Event Triggers 32a' and 32b' provide notice to the computer system 20 of remote command completion by triggering the Events 36a and 36b. Since the Event Triggers 32a' and 32b' may provide notice to the computer system 20 after a potentially disruptive network communication 34 has occurred, remote command execution may still complete even if the network connections do not remain persistent. Thus, remote command performance is improved over conventional systems.

Turning now to Fig. 2, a client/server model 10b illustrates an exemplary system in accordance with an aspect of the present invention, wherein a client 40 initiates a remote command on a server 42 which may also be referred to as remote system 42. It is to be appreciated that a plurality of remote commands may be executed on a plurality of systems and that commands may also be directed from the server 42 to the client 40 and/or other systems (not shown).

Before executing a remote command, the client 40 may generate an identifier 44 for a completion event 46 that may be fired (*e.g.*, triggered) when remote execution of a command completes. For example, the client 40 may configure an event subscription 48 (*e.g.*, Windows Management Infrastructure (WMI) event) that may be triggered by the completion event 46. Furthermore, the event subscription 48 may also be employed to process the results of the remote command execution. The client 40 may then acquire an interface pointer 50 to an object 52 and invoke a remote object call 54 *via* an object interface 56. The remote object 52 may provide a data structure and object functions for carrying out remote command execution wherein the object interface 56 may provide access to the remote object 52. An exemplary object interface 56 is illustrated below.

Object Interface (e.g.,) IextensibleClusterCmd

```

5  {
  a)      HRESULT Execute( [in] BSTR bstrCLSID,      //CLSID of component
                                //implementing desired
                                functionality
  b)                                [in] BSTR bstrEventServer, //name of server completion
                                //event is to be fired on
10  c)                                [in] BSTR bstrEventGuid, //GUID of completion event to
                                fire
  d)                                [in] BSTR bstrInputArg,  //input argument to be passed
                                to
15  e)                                //component
                                [in] BSTR bstrUser,          //username to employ when
                                calling
                                //component
  f)                                [in] BSTR bstrDomain,     //domain for bstrUser
  g)                                [in] BSTR bstrPwd );       //password for bstrUser
20  };

```

It is to be appreciated that the object interface 56 may be implemented *via* a distributed object architecture for communicating with remote objects (e.g., COM, DCOM, CORBA, etc.) The object interface 56 may include: an identification of the object 52 to be invoked (e.g., line (a) above), a name of a server on which the completion event 46 should be fired (e.g., line (b) above) [It is noted the name may generally be the same as the machine on which the client 40 is running, but need not be], the identifier 44 (e.g., line (c) above) to be attached to the completion event 46, the input argument (e.g., line (d) above) for the remote object 52 and/or security information (e.g., lines (e, f and g) above) specifying a user context the remote object 52 should be invoked as. As an example, a remote object call 54 may be invoked by the command

IExtensibleClusterCmd::Execute() and passing the exemplary arguments described above in lines a through f. If the IExtensibleClusterCmd::Execute() returns successfully, the client 40 may release the interface pointer 50 and proceed to perform other work. The client may then be notified automatically *via* the event subscription 48 when and/or if remote execution completes.

After the remote object call 54 has been invoked, the remote system 42 may then proceed to perform remote execution of the command. Invoking a call to the object interface 56 on the remote system 42 may result in a work item 60 being queued to a thread pool 64 (*e.g.*, *via* Windows NT 5 thread pool APIs) thereby enabling the call to return at once. The work item 60 may then be processed by a thread 68 from the thread pool 60. The object 52 with the requested identification parameter (*e.g.*, specified in line a above) may then be instantiated by the thread 60 and queued for execution 70 *via* a work interface described below. The input argument (*e.g.*, specified in line d above) may also be passed to the thread 60 for object execution. Below is an exemplary work interface for executing the remote object 52 in accordance with the present invention.

```
work interface (e.g.,) IClusterWork
{
    HRESULT Execute( [in] BSTR bstrInputArg,
                    [out] BSTR *pbstrOutputArg );
};
```

When the above work interface call returns, an HRESULT return status, together with the output argument from the call (*e.g.*, in the pbstrOutputArg parameter) may be marshalled into the completion event 46 (*e.g.*, of type ICEClusterActivity : the ActivityStatus property of the event 46 may be set to the HRESULT of calling IClusterWork::Execute()). An ActivityOutput property of the completion event 46 may also contain data returned in the pbstrOutputArg argument to the IClusterWork::Execute() with the event's ActivityGUID property set to the GUID parameter passed in the bstrEventGuid argument (*e.g.*, line c above) in the call to the object interface 56. The completion event 46 may then be fired on the machine (*e.g.*, the client 40) specified in the bstrEventServer argument (*e.g.*, line b above). Since the completion event 46 may be fired after remote command execution, and hence any network disruption 34 is therefore likely complete, the present invention mitigates the problem faced by conventional remote invocation technologies that rely on persistent network connections.

The interfaces described above provide an advantageous extensibility mechanism to provide new system functionality. Extensibility may be achieved by employing an object that implements the work interface (*e.g.*, IClusterWork), pass input data the work interface requires in the input argument parameter of the object interface and marshall the output data into the work interface output argument (*e.g.*, pbstrOutputArg). In this manner, new functionality may be invoked on a remote machine by installing a component on the remote machine implementing the desired functionality and specifying the component as part of the method invocations described above.

Referring now to Fig. 3, a methodology illustrates a particular aspect of the present invention relating to parallel asynchronous command execution. Initial aspects of the command execution are depicted in relation to a client side execution 10c and a server side execution 10d. Particular aspects of the client and server side execution will be described in more detail below in relation to Figs. 4 and 5.

At step 100, an event identifier may be generated by the client to associate a name for a server to respond after completion of the remote command execution. For example, a string may be employed associating an event with a GUID as described above in relation to line (c) of the object interface. At step 102, the client may configure an event subscription for receiving notice and results of completed and/or unsuccessful remote command execution. For example, the event subscription may be configured according to Microsoft's Windows Management Infrastructure (WMI). At step 104, an object interface pointer is obtained from the server wherein the client may be directed to a remote object providing the remote command execution. At step 106, the client invokes the remote command by calling the server and passing arguments (*e.g.*, input buffers, names, identifiers) to the object interface described above.

Proceeding to step 108, the server receives the call and arguments generated by the client. At step 110, the server attempts to queue a work item for processing the received call from step 108. At step 112, a determination is made as to whether the work item was queued successfully (*e.g.*, whether server has enough resources to process requested call). If the work item was queued successfully at step 112, the process proceeds to notify the client at step 114 (*e.g.*, *via* a return success flag) and the process

proceeds to step 118. If the work item was not queued successfully at step 112, the client may be similarly notified of a failure at step 116 and the process then proceeds to step 120. If the client was notified of the work item being queued successfully at step 118, the client may perform other tasks while waiting for the remote command to complete wherein the client may then be notified of completion *via* a completion event as described above. If the client was notified that a work item was not successfully queued at step 120, the subscription event described above may then be canceled and/or other processing may commence such as notifying a user of the unsuccessful attempt by the server to begin the requested work item.

Turning now to Fig. 4, the server side methodology 10d illustrated in Fig. 3 is shown in more detail in relation to server side processing after a work item as been successfully queued in step 112. At step 122, the work item is taken off of the queue for processing (*e.g.*, thread processing). At step 124, the object component specified in the remote command invocation described above is instantiated for further processing.

At step 126, a determination is made as to whether the instantiation of step 124 was successful. If instantiation was successful at step 126, the process proceeds to step 128 wherein a method is called to execute the object component functionality. At step 130, a completion event is created wherein results of the method execution are provided upon completion of the method execution. Upon completion, the completion event then triggers an event at step 132 to notify the client of the method completion and to provide the client with results of the method execution.

If instantiation was not successful at step 126 above, the process proceeds to step 134. At step 134, a completion event may be created wherein failure information of the unsuccessful object instantiation of step 126 is provided. The process then proceeds to step 132 wherein the client may be notified and provided with the failure information *via* the event trigger described above.

Now referring to Fig. 5, the client methodology illustrated in Fig. 3 is shown in more detail in relation to client side processing after steps 118 and/or step 120 wherein the server has caused the event configured in step 102 to complete. At step 140, the client is notified of the completed remote command execution *via* the subscription event

configured at step 102 of Fig. 3. In this manner, the client may execute a plurality of remote asynchronous commands without waiting for completion of the commands on the remote machines. Moreover, since the client is notified after completion of the remote command, network disruptions that may affect conventional invocation technologies are mitigated. At step 144, the client may then extract execution status and results associated with the remote command execution. At step 148, the client may then perform related processing associated with the results and/or status of step 148. This processing may include, for example, signaling an error *via* a user interface if error results were obtained from step 134 of the server side execution. At step 152, the event subscription configured at step 102 may then be cancelled upon notification and results of the remote command execution.

In order to provide a context for the various aspects of the invention, Fig. 6 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the various aspects of the present invention may be implemented. While the invention has been described above in the general context of computer-executable instructions of a computer program that runs on a computer, those skilled in the art will recognize that the invention also may be implemented in combination with other program modules. Generally, program modules include routines, programs, components, data structures, *etc.* that perform particular tasks and/or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the inventive methods may be practiced with other computer system configurations, including single-processor or multiprocessor computer systems, minicomputers, mainframe computers, as well as personal computers, hand-held computing devices, microprocessor-based or programmable consumer electronics, and the like. The illustrated aspects of the invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. However, some, if not all aspects of the invention can be practiced on stand-alone computers. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to Fig. 6, an exemplary system for implementing the various aspects of the invention includes a conventional server and/or client computer 220, including a processing unit 221, a system memory 222, and a system bus 223 that couples various system components including the system memory to the processing unit 221.

- 5 The processing unit may be any of various commercially available processors, including but not limited to Intel x86, Pentium and compatible microprocessors from Intel and others, including Cyrix, AMD and Nexgen; Alpha from Digital; MIPS from MIPS Technology, NEC, IDT, Siemens, and others; and the PowerPC from IBM and Motorola. Dual microprocessors and other multi-processor architectures also may be employed as
10 the processing unit 221.

- The system bus may be any of several types of bus structure including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of conventional bus architectures such as PCI, VESA, Microchannel, ISA and EISA, to name a few. The system memory includes read only memory (ROM) 224 and random
15 access memory (RAM) 225. A basic input/output system (BIOS), containing the basic routines that help to transfer information between elements within the server computer 220, such as during start-up, is stored in ROM 224.

- The server computer 220 further includes a hard disk drive 227, a magnetic disk drive 228, *e.g.*, to read from or write to a removable disk 229, and an optical disk drive
20 230, *e.g.*, for reading a CD-ROM disk 231 or to read from or write to other optical media. The hard disk drive 227, magnetic disk drive 228, and optical disk drive 230 are connected to the system bus 223 by a hard disk drive interface 232, a magnetic disk drive interface 233, and an optical drive interface 234, respectively. The drives and their associated computer-readable media provide nonvolatile storage of data, data structures,
25 computer-executable instructions, etc. for the server computer 220. Although the description of computer-readable media above refers to a hard disk, a removable magnetic disk and a CD, it should be appreciated by those skilled in the art that other types of media which are readable by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, and the like, may also be used in
30 the exemplary operating environment, and further that any such media may contain

computer-executable instructions for performing the methods of the present invention.

A number of program modules may be stored in the drives and RAM 225, including an operating system 235, one or more application programs 236, other program modules 237, and program data 238. The operating system 235 in the illustrated
5 computer may be a Microsoft operating system (*e.g.*, Windows operating system, NT server). It is to be appreciated that other operating systems may be employed (*e.g.*, Unix operating system).

A user may enter commands and information into the server computer 220 through a keyboard 240 and a pointing device, such as a mouse 242. Other input devices
10 (not shown) may include a microphone, a joystick, a game pad, a satellite dish, a scanner, or the like. These and other input devices are often connected to the processing unit 221 through a serial port interface 246 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, a game port or a universal serial bus (USB). A monitor 247 or other type of display device is also connected to the system bus 223 via an
15 interface, such as a video adapter 248. In addition to the monitor, computers typically include other peripheral output devices (not shown), such as speakers and printers.

The server computer 220 may operate in a networked environment using logical connections to one or more remote computers, such as a remote client computer 249. The remote computer 249 may be a workstation, a server computer, a router, a peer device or
20 other common network node, and typically includes many or all of the elements described relative to the server computer 220, although only a memory storage device 250 is illustrated in FIG. 5. The logical connections depicted in FIG. 6 may include a local area network (LAN) 251 and a wide area network (WAN) 252. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets
25 and the Internet.

When employed in a LAN networking environment, the server computer 220 may be connected to the local network 251 through a network interface or adapter 253. When utilized in a WAN networking environment, the server computer 220 generally may include a modem 254, and/or is connected to a communications server on the LAN,
30 and/or has other means for establishing communications over the wide area network 252,

such as the Internet. The modem 254, which may be internal or external, may be connected to the system bus 223 *via* the serial port interface 246. In a networked environment, program modules depicted relative to the server computer 220, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that
5 the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

In accordance with the practices of persons skilled in the art of computer programming, the present invention has been described with reference to acts and symbolic representations of operations that are performed by a computer, such as the
10 server computer 220, unless otherwise indicated. Such acts and operations are sometimes referred to as being computer-executed. It will be appreciated that the acts and symbolically represented operations include the manipulation by the processing unit 221 of electrical signals representing data bits which causes a resulting transformation or reduction of the electrical signal representation, and the maintenance of data bits at
15 memory locations in the memory system (including the system memory 222, hard drive 227, floppy disks 229, and CD-ROM 231) to thereby reconfigure or otherwise alter the computer system's operation, as well as other processing of signals. The memory locations wherein such data bits are maintained are physical locations that have particular electrical, magnetic, or optical properties corresponding to the data bits.

20 What has been described above are preferred aspects of the present invention. It is, of course, not possible to describe every conceivable combination of components and/or methodologies for purposes of describing the present invention, but one of ordinary skill in the art will recognize that many further combinations and permutations of the present invention are possible. Accordingly, the present invention is intended to
25 embrace all such alterations, modifications and variations that fall within the spirit and scope of the appended claim

Claims

What is claimed is:

1. A system for parallel asynchronous command execution, comprising:
a first computer system for directing a call to invoke a remote procedure in a second computer system, the first computer and second computer communicating *via* a non-persistent connection;
wherein the second computer system upon completion of the remote procedure generates an event trigger and transmits the event trigger and remote procedure results to the first computer system.
2. The system of claim 1 further comprising a distributed object architecture for communicating between the first computer system and the second computer system.
3. The system of claim 2 wherein the distributed object architecture is implemented *via* at least one of COM, DCOM, and CORBA interface languages.
4. The system of claim 1 wherein the first computer system configures an event to receive the remote procedure results from the second computer system.
5. The system of claim 4 wherein the event is a Windows Management Infrastructure event.
6. The system of claim 4 wherein the event is provided with an identifier for enabling the second computer system to notify the first computer system.
7. The system of claim 1 further comprising a work item and a thread for processing the remote procedure.

8. The system of claim 1 further comprising a completion event on the second computer system for notifying the first computer system.

9. The system of claim 1 further comprising an object interface for providing remote access between the first computer system and the second computer system.

10. The system of claim 9 wherein the object interface further includes an identification for a remote object.

11. The system of claim 9 wherein the object interface further includes a computer name for identifying where to trigger an event.

12. The system of claim 9 wherein the object interface further includes an identification for an event for the second computer system to trigger.

13. The system of claim 9 wherein the object interface further includes an input argument for providing results from the remote procedure.

14. The system of claim 9 wherein the object interface further includes at least one of a username, domain, and password for specifying a user context for the remote procedure.

15. The system of claim 1 wherein the second computer system further comprises a work interface for executing the remote procedure.

16. The system of claim 15 wherein the work interface further comprises an input argument and an output argument.

17. A system for parallel asynchronous command execution, comprising:
a first computer system for directing at least one call to invoke a remote procedure in at least one other computer system, the at least one other computer system upon completion of the remote procedure establishing a non-persistent connection to the first computer system, the at least one other computer system generating an event trigger and transmitting the event trigger and remote procedure results to the first computer system.

18. A method for parallel asynchronous command execution, comprising the steps of:

directing at least one call from a first computer system to invoke a remote procedure in at least one other computer system;

establishing a non-persistent connection between the first computer system and the at least one other computer system upon completion of the remote procedure; and

generating an event trigger and transmitting the event trigger and remote procedure results to the first computer system.

19. The method of claim 18 further comprising the step of communicating between the first computer system and the at least one other computer system *via* a distributed object architecture.

20. The method of claim 19 wherein the distributed object architecture is implemented *via* at least one of COM, DCOM, and CORBA interface languages.

21. The method of claim 18 further comprising the step of configuring an event on the first computer system to receive results from the at least one other computer system.

22. The method of claim 21 wherein the event is a Windows Management Infrastructure event.

23. A system for parallel asynchronous command execution, comprising:
means for directing at least one call from a first computer system to invoke a remote procedure in at least one other computer system;
means for establishing a non-persistent connection between the first computer system and the at least one other computer system upon completion of the remote procedure; and
means for generating an event trigger and transmitting the event trigger and remote procedure results to the first computer system.

24. A system for parallel asynchronous command execution, comprising:
a server for responding to at least one remote call by invoking a remote procedure;
the server establishes a non-persistent connection to communicate results of the remote procedure; and
the server generates an event trigger and transmits the event trigger and remote procedure results upon completion of the remote procedure.

25. The system of claim 24 wherein a client receives the event trigger and remote procedure results *via* the non-persistent connection.

26. The system of claim 25 wherein the client configures an event to receive the remote procedure results.

27. A system for parallel asynchronous command execution, comprising:
a first computer for directing a call to invoke a remote procedure in a second computer, the first computer transmits a non-persistent signal to the second computer;
wherein the second computer upon completion of the remote procedure generates an event trigger and transmits the event trigger and remote procedure results to the first computer *via* the signal.

28. A computer-readable medium having computer-readable instructions for performing the acts of, comprising:

responding to at least one remote call by invoking a remote procedure;

establishing a non-persistent connection to communicate results of the remote procedure; and

generating an event trigger and transmitting the event trigger and remote procedure results upon completion of the remote procedure.

29. The computer-readable medium of claim 28 further comprising, receiving the event trigger and remote procedure results *via* the non-persistent connection.

30. The computer-readable medium of claim 28 further comprising, configuring an event to receive the remote procedure results.

Abstract of the Invention

A system for parallel asynchronous command execution is provided. The system includes a first computer system for directing a plurality of calls to at least one other computer system. The first computer system invokes a remote procedure in the other computer system and receives results from the procedure *via* an event triggered by the other computer system. The system may also include a distributed object architecture for communicating between the first computer system and the other computer system.

\\SBSTPA\shared\HAM\MSET\PI21US\MS150658.1v5.doc

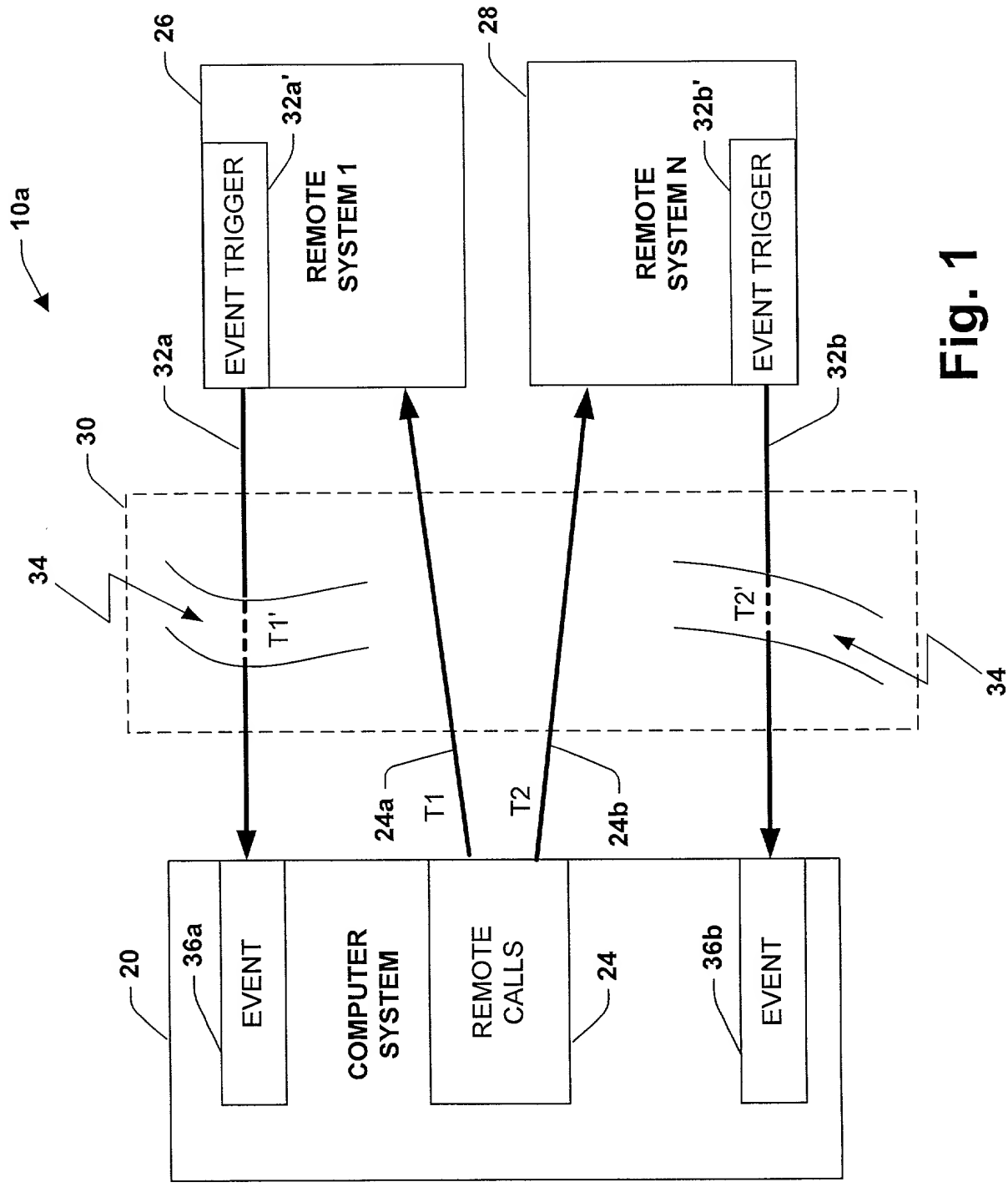


Fig. 1

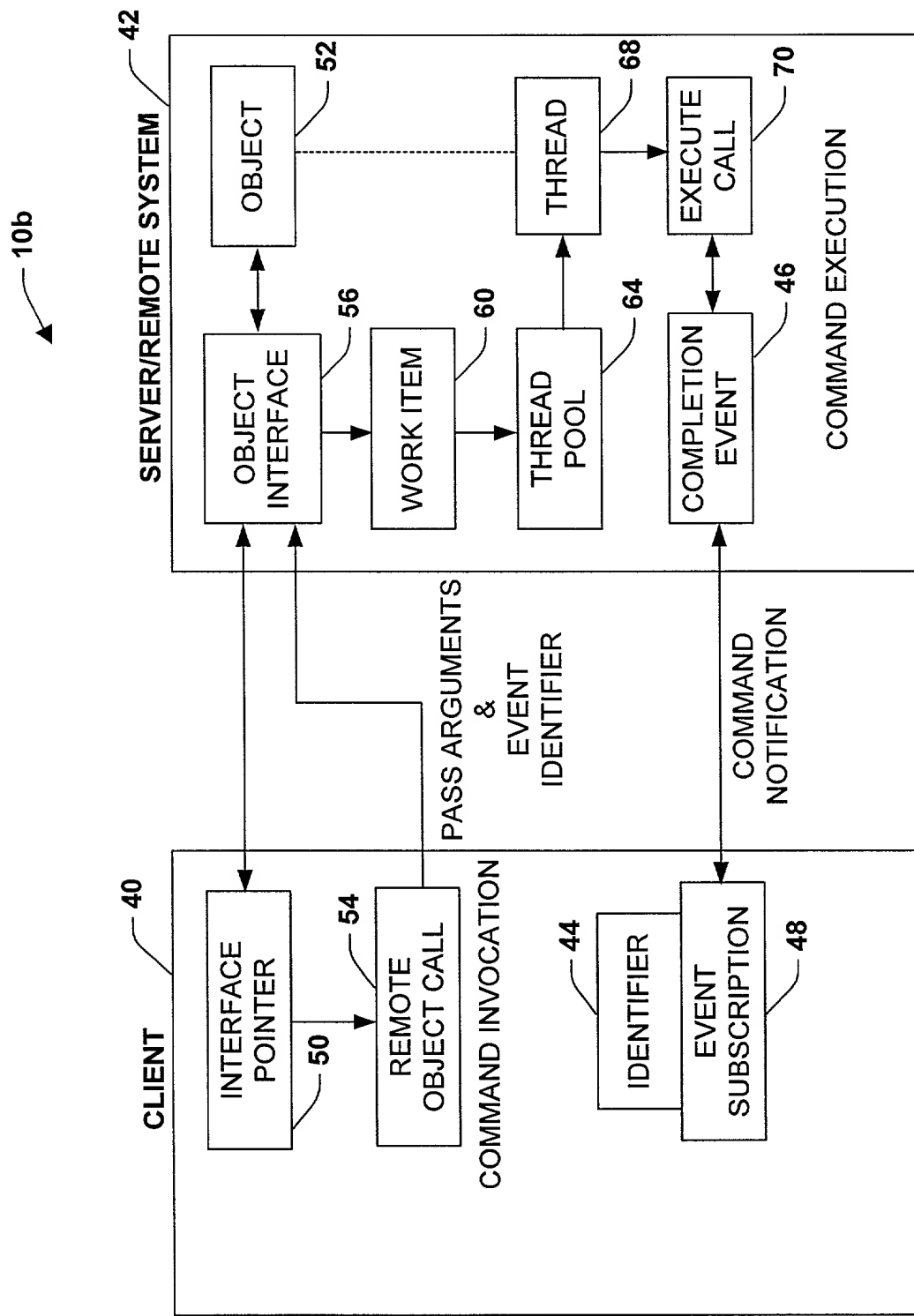


Fig. 2

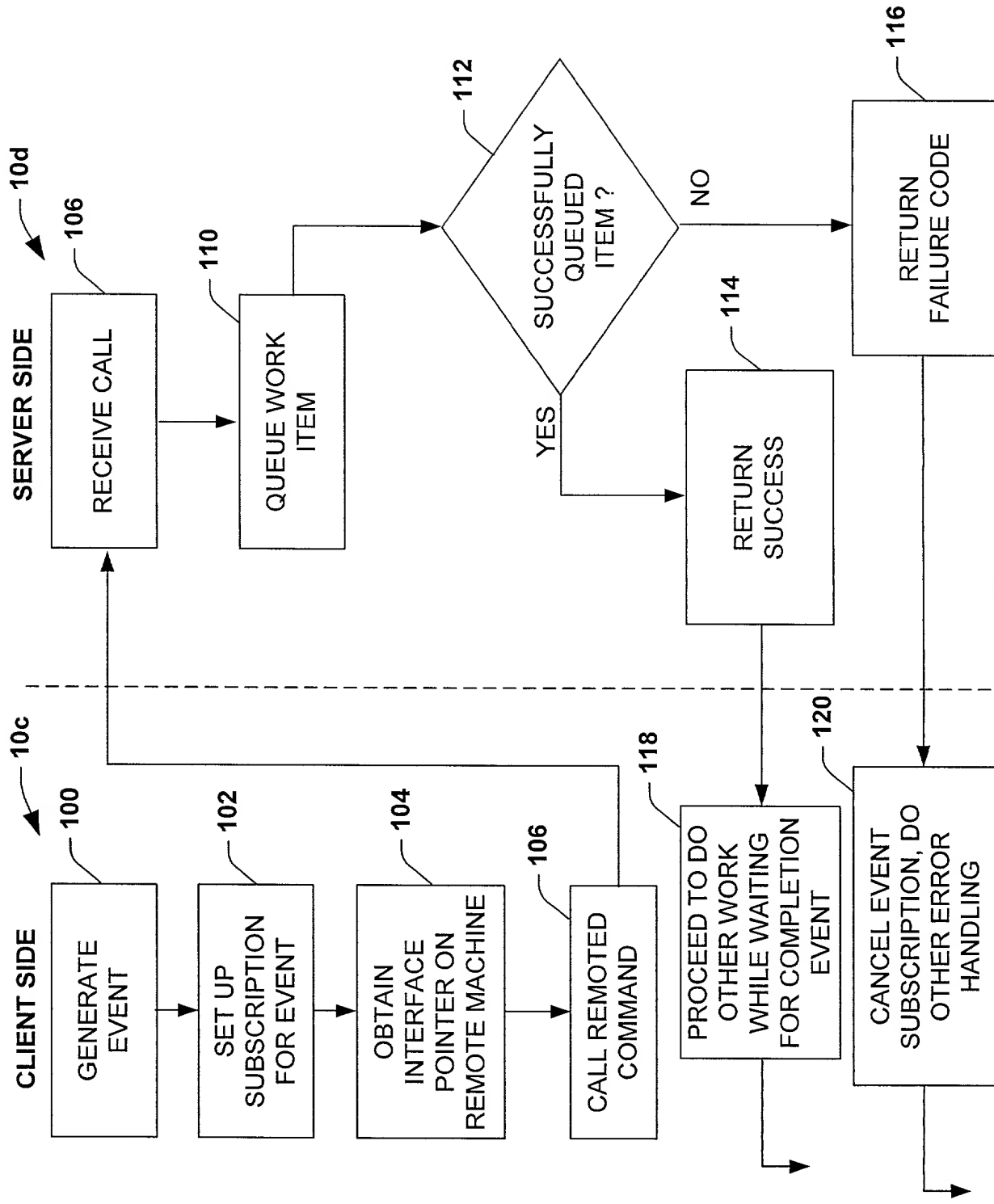


Fig. 3

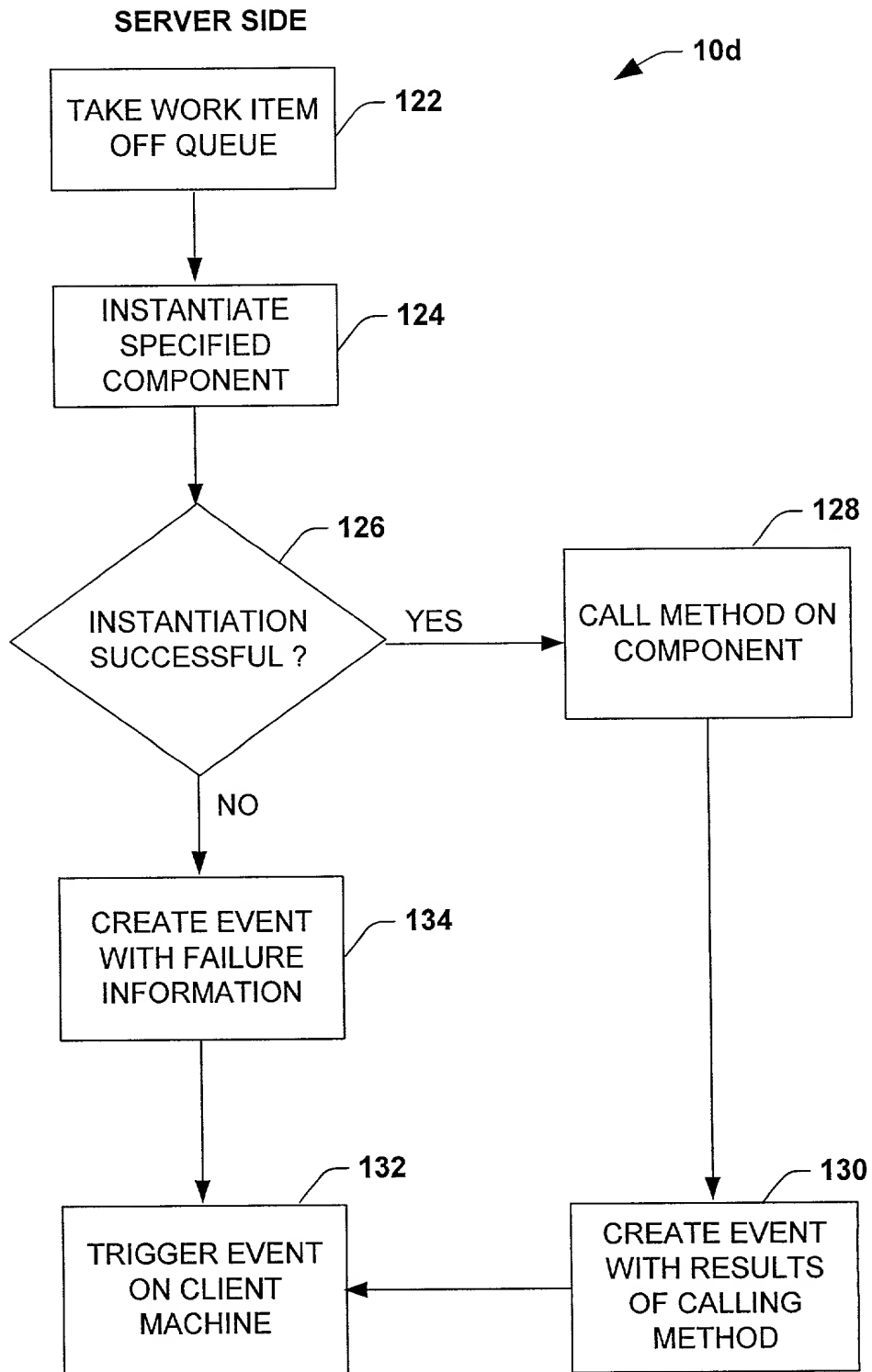


Fig. 4

CLIENT SIDE

10c

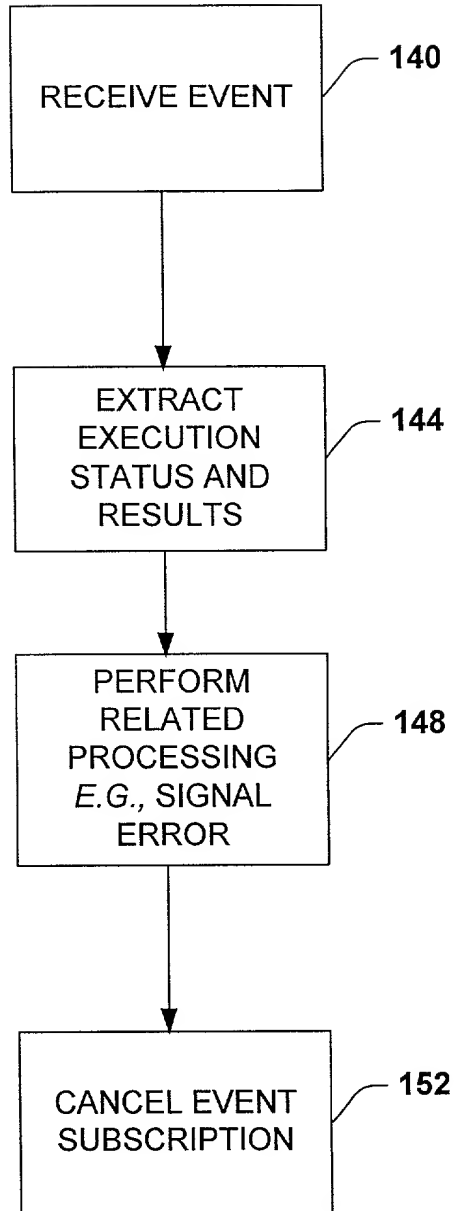


Fig. 5

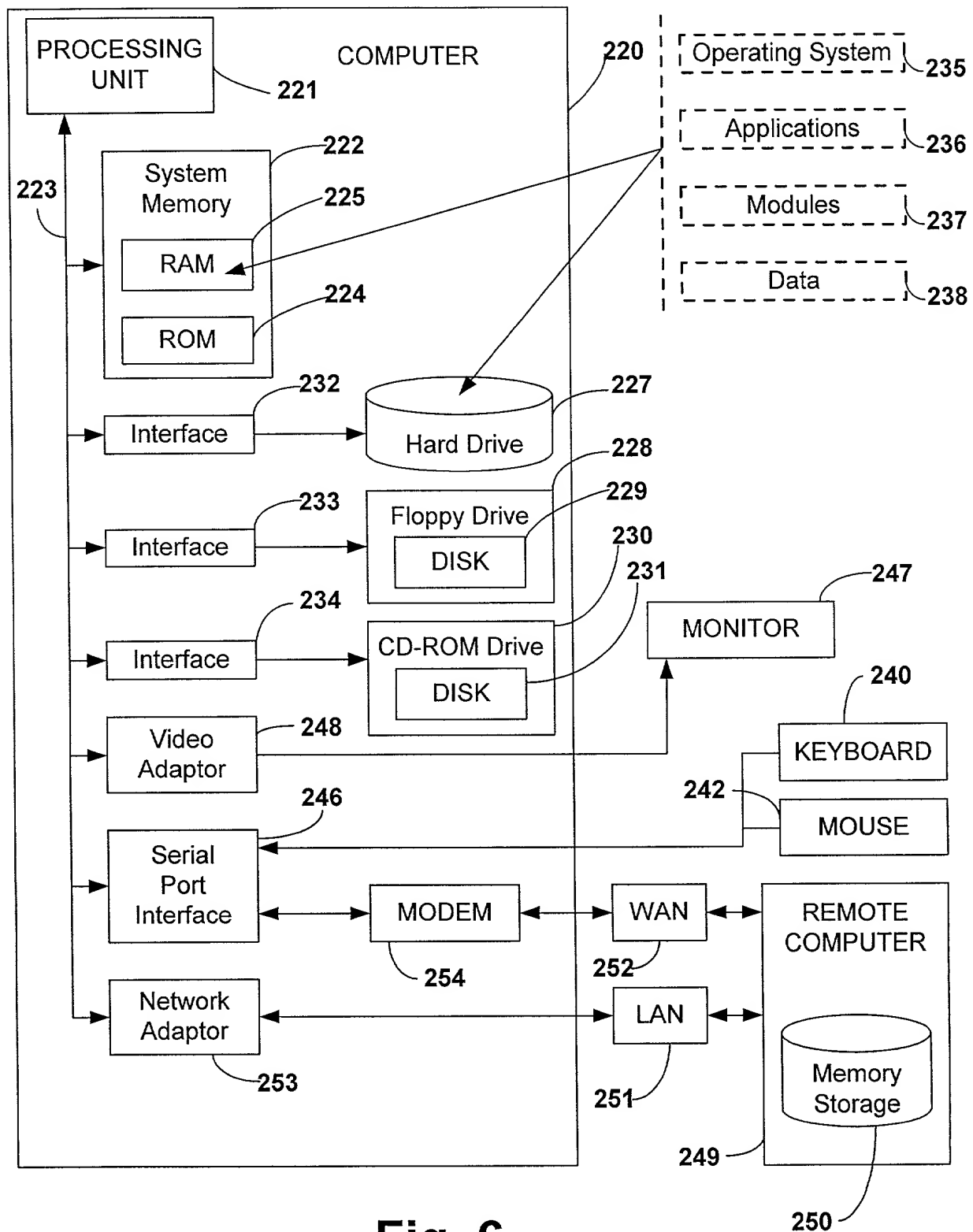


Fig. 6

COMBINED DECLARATION AND POWER OF ATTORNEY
(ORIGINAL, DESIGN, NATIONAL STAGE OF PCT)

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name, I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled: **SYSTEM AND METHOD FOR PARALLEL ASYNCHRONOUS**

EXECUTION OF COMMANDS

the specification of which

- (a) X is attached hereto.
 (b) was filed on as Serial No. or
 Express Mail No. , as Serial No. not yet known, and was amended on
 (if applicable).
 (c) was described and claimed in PCT International Application No. filed
 on and amended under PCT Article 19 on (if any).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to patentability in accordance with Title 37, Code of Federal Regulations '1.56(a).

PRIORITY CLAIM

I hereby claim foreign priority benefits under Title 35, United States Code, '119 of any foreign application(s) for patent or inventor's certificate or of any PCT international application(s) designating at least one country other than the United States of America listed below and have also identified below any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date before that of the application(s) of which priority is claimed.

- (d) X no such applications have been filed.
 (e) such applications have been filed as follows.

**EARLIEST FOREIGN APPLICATION(S), IF ANY FILED WITHIN 12 MONTHS
(6 MONTHS FOR DESIGN) PRIOR TO THIS U.S. APPLICATION**

COUNTRY	APPLICATION NUMBER	DATE OF FILING (day, month, year)	PRIORITY CLAIMED UNDER 35, USC 119
<u> </u>	<u> </u>	<u> </u>	<u> </u> Yes <u> </u> No
<u> </u>	<u> </u>	<u> </u>	<u> </u> Yes <u> </u> No
<u> </u>	<u> </u>	<u> </u>	<u> </u> Yes <u> </u> No

**ALL FOREIGN APPLICATION(S), IF ANY FILED MORE THAN 12 MONTHS
(6 MONTHS FOR DESIGN) PRIOR TO THIS U.S. APPLICATION**

POWER OF ATTORNEY

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith. (List name and registration number)

Himanshu S. Amin, Reg. No. 40,894; Gregory Turocy, Reg. No. 36,952;
Christopher P. Harris, Reg. No. 43,660; Eric M. Highman,
Reg. No. 43,672; and Gary J. Pitzer, Reg. No. 39,334.

Katie E. Sako, Reg. No. 32,628 and Daniel D. Crouse, Reg. No. 32,022.

The undersigned to this declaration and power of attorney hereby authorizes the U.S. attorney(s) named herein to accept and follow instructions from:

Name(s) of authorized representative(s) _____
Address _____

as to any actions to be taken in the Patent and Trademark Office regarding this application without direct communication between the U.S. attorney(s) and the undersigned. In the event of a change in the person(s) from whom instructions may be taken, the U.S. attorney(s) will be so notified by the undersigned.

Send Correspondence To:

Himanshu S. Amin
AMIN, ESCHWEILER & TUROCY, LLP
24TH Floor, National City Center
1900 East 9TH Street
Cleveland, Ohio 44114

Direct Telephone Calls To:
(name and telephone number)

Himanshu S. Amin
(216) 696-8730

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued therein.

Full name of sole or first inventor, if any: Alexander E. Mallet
Inventor's signature: Alexander E. Mallet
Date: 6/9/2000 Country of Citizenship: German
Residence: Bellevue, Washington
Post Office Address: 1070 W. Lk Sammanish Parkway NE
Bellevue, Washington 98008

Full name of second or joint inventor, if any: Justin Grant
Inventor's signature: Justin Grant
Date: 6-9-00 Country of Citizenship: U.S.
Residence: Seattle, Washington
Post Office Address: 10 Smith Street 3205 S. Dearborn St
Seattle, Washington 98144

CHECK FOR ANY OF THE FOLLOWING ADDED PAGE(S) WHICH
FORM A PART OF THIS DECLARATION

X Signature for third and subsequent joint inventors. Number of pages added 1.

Full name of third joint inventor, if any: Michael W. Thomas
Inventor's signature: Michael W. Thomas
Date: June 9, 2000 Country of Citizenship: U.S.
Residence: Seattle, Washington
Post Office Address: 822 22nd Avenue 7134 166th Ave. SE
Seattle, Washington 98122 Bellevue, WA 98008

X This declaration ends with this page.